REMARKS

Claims 1-25 are pending in the application. Applicants gratefully acknowledge Examiner's indication that claims 9-25 are in condition for allowance.

Claims 1-8 stand rejected under 35 U.S.C. § 103 as being unpatentable over U.S. Patent No. 5,287,309 to Kai in view of U.S. Patent No. 6,167,488 to Koppala. Applicants respectfully traverse the rejections and earnestly request reconsideration of the rejections for the following reasons.

To establish a *prima facie* case of obviousness based on the combination of Kai and Koppala, various criteria must be met. For instance, the combination *must* teach or suggest all the claim limitations. Further, there must be some suggestion or motivation in the references or in the knowledge generally available to one skilled in the art to combine their teachings. The teaching or suggestion to make the claimed combination must both be found in the prior art and not based on *hindsight* in view of applicant's disclosure (see, e.g., MPEP 2141, 2143, 2143.03).

Here, it is respectfully submitted that at the very minimum, the combination of Kai and Koppala is legally deficient to establish a *prima facie* case of obviousness against claims 1, 2 and 6, for at least the following reasons: (1) there is no motivation or suggestion to combine the teachings of Kai and Koppala to derive the claimed inventions; and (2) even if Kai and Koppala are combined, such combination does not teach or suggest all the elements of claims 1, 2 and 6.

More specifically, with respect to (1), it is respectfully submitted that the structures and functions of the stack circuits of Kai and Koppala are so fundamentally different from those of the claimed inventions, that one of ordinary skill in the art would not be motivated to combine the teachings of Kai and Koppala to derive the claimed inventions.

The inventions of claims 1, 2 and 6 are directed, in general, to devices having a stack architecture wherein a stack comprises a *plurality of banks* and wherein each bank has a corresponding *bank pointer*. Such framework enables the performance of *multi-word push* and *multi-word pop* stack operations using the multi-bank framework. In other words, the claimed inventions enable multi-word stack operations using a multi-bank stack framework and a plurality of bank pointers that store address data for corresponding banks in the stack.

In general, Kai is directed to a stack memory framework that is capable of simultaneously executing *Push* (Write) and a *Pop* (read) operations, with respect to an address in a Stack Pointer. The Kai framework enables both *Push* and *Pop* operations to be simultaneously performed before deciding which operation (either *Push* or *Pop*) is to be selected, which results in high-speed processing. Ultimately, however, only one of the single-word stack accesses (e.g., *Push* or *Pop*) is validated by updating the Stack Pointer. Therefore, using the Kai framework, a Last-in-First-Out (LIFO) stack may store invalid data in stack addresses starting from the address designated by the Stack Pointer and higher, which is inconsequential because of the updating of the Stack pointer which validates the selected operation (see, Kai, Col. 2, lines 56-68).

More specifically, FIG. 1 of Kai discloses an embodiment using a two-port RAM, which enables simultaneous read and write operations by design, as is understood by those of ordinary skill in the art. In such embodiment, the stack pointer (2) holds an address **N** corresponding to *Push* data, and when an access operation is commenced, simultaneous *Push* and *Pop* operations are performed, whereby data is written (push) to the address **N** (as specified by the Stack Pointer) and data is read (pop) from the address **N-1**. If the *Push* operation is ultimately selected, the address **N** in the Stack Pointer (2) is incremented by 1 (via incrementer (3)) to validate the *Push* operation. On the other hand, if the *Pop* operation is selected, the address **N-1** is stored in the

Stack Pointer (2) to validate the *Pop* operation (in which case the *Push* operation, by which data was stored at address **N**, is invalid, and of no consequence because the Stack pointer stores address **N-1**) (See, Col. 3, line 30 – Col. 4, line 9).

The embodiment illustrated in FIG. 2 of Kai is similar in concept to the embodiment of FIG. 1, except that FIG. 2 discloses a stack memory (RAM) comprising an even and odd bank (10, 11), wherein each bank comprises an independent one-port RAM (see Col. 4, lines 27-30). As is understood by those of ordinary skill in the art, a single-port RAM cannot be used for performing simultaneous read and write operations. *Therefore, in FIG. 2, the use of a two-bank memory with two independent single-port RAMs is* ***necessary*** *to enable simultaneous Push and Pop operations of the stack memory,* as is desired in Kai to achieve high-speed performance.

In the embodiment of FIG.. 2, one bank is used for the *Push* operation and the other bank is used for the *Pop* operation, depending on whether a stack pointer (6) holds and even or odd address (Kai, Col. 4, lines 14-20, lines 50-57). Essentially, the Stack Pointer (6) holds an address **SP** over all of the stack memory for indicating data to be processed by the *Pop* operation (whereas SP is incremented by one (**SP+1**) which indicates the address to which data is stored for the *Push* operation) (see, Col. 4, lines 58-62). The embodiment of FIG. 2 comprises control circuitry to control the simultaneous read and write operations and then update the Stack pointer (6) to validate the desired operation (see, Col. 5, line 5 – Col. 6, line 10). Ultimately, when the two-bank stack framework of FIG. 2 is viewed as one stack unit, regardless of whether an even or odd address is in the Stack Pointer (6), a simultaneous *Pop* and *Push* operation results in data being Read (*Pop*) from the address **SP** as specified by the stack pointer (6), and data being written (*Push*) to an address specified by **SP+1** (see, Col. 6, lines 11-19, and lines 50-58). If the *Push* operation is ultimately selected, the address of the Stack Pointer (6) is incremented by "1" to

validate the *Push* operation. On the other hand, if the *Pop* operation is selected, the address of the stack pointer (6) is decremented by "1", so as to validate the *Pop* operation (see Col. 6, lines 20-33 and lines 59-62).

Therefore, although Kai discloses a two-bank stack memory, it is abundantly clear the Kai framework is implemented with independent, single-port memory devices to simultaneously perform a *single-word Pop* operation in one bank and a *single-word Push* operation in the other bank, wherein only one of such operations is ultimately validated. Kai does not disclose or suggest a two-bank framework in which a *multi-word pop* operation can be performed by reading a word from each bank or a *multi-word Push* operation can performed by writing a word to each bank. Indeed, not only is the control circuitry in FIG. 2 unsuitable for such tasks, such operation would be inconsistent with the intent and purpose of Kai , which is to perform simultaneous Push and Pop operations using respective banks, to provide high-speed processing.

Furthermore, Koppala does not cure the deficiencies of Kai in this regard. Although Koppala arguably discloses performing multi-word stack operations, Koppala implements a *multi-port memory device* to perform multiple read and/or write operations using different ports. For instance, FIG. 12 of Koppala is a circuit that generates pointers for reading two data words from the stack (255) (Fig. 3) and writing one data word data to the stack (255) using two separate read ports (340) and (350) and one write port (330). Koppala does not disclose or suggest a multi–bank framework for reading or writing multi-word data. Therefore, even if Koppala and Kai are combined, there is no teaching or suggestion by such combination of a multi-bank framework that enables multi-word stack operations, as essentially claimed in claims 1, 2 and 6 of the present invention.

Examiner essentially contends on page 3 of the Office Action that it would have been obvious to one of ordinary skill in the art at the time of the invention to have provided the circuit of Kai with capability to remove a two-word item from the stack as taught or suggested by Koppala, because it would have provided a mechanism to execute either the two Push or two POP operations (instead of one Push and one Pop in the different banks) on the stack, thus making the stack operation much faster. *Applicants respectfully disagree with such contention for the following reasons.*

First, as noted above, there is absolutely no teaching or suggestion in Kai of using the two-bank framework (Fig. 2) to perform a multi-word Pop or Push operations by simultaneously reading/writing a word from/to each bank, *because the conceptual premise behind Kai is to perform simultaneous Push and Pop operations with respective banks, and subsequent validation of one of such operations, to provide high-speed processing*. Examiner's contention that "variation of the simultaneous Pop and Push can be implemented ... by simple mathematical transformation" is misplaced, because the stack control circuitry is structured for performing simultaneous read and write operations in the different banks and validating a selected operation, and such circuitry would require modification, not just simple mathematical changes, to perform other types of operations, such as a two-word push operation using both banks..

Furthermore, one of ordinary skill in the art would not look to the teachings of Koppala to modify the circuit in FIG. 2 of Kai to perform multi-word operations in each bank. Indeed, as explained above, Koppala teaches a framework in which multi-word operations are performed by simultaneously reading/writing data using different ports of a multi-port stack memory. In contrast, as noted above, Kai specifically discloses that each bank (10, 11) in FIG. 2 comprises a

single-port RAM memory. As such, these frameworks are incompatible and one of ordinary skill in the art would not be motivated to combine the teachings.

Moreover, even if the teachings of Koppala could somehow be combined with the embodiment of FIG. 1 of Kai, which uses a multi-port RAM, this would not result in a multi-bank stack memory. Therefore, based at least on the above, there is simply no teaching or suggestion to combine Koppala and Kai.

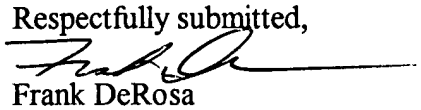
Furthermore, even assuming, arguendo, that Koppala and Kai were properly combinable, the combination of Kai and Koppala would fail to establish prima facie case of obviousness because such combination does not disclose or suggest a circuit having *a bank pointer associated with each bank of a stack, wherein each bank pointer stores address data for its respective bank*, as essentially claimed in claims 1, 2 and 6. In contrast, Kai discloses only a single stack pointer (6) (see Fig. 2) that stores an address over all of the stack memory (see, Col. 4, lines 58-60). The stack pointer (6) does not comprise a plurality of bank pointers that store the bank addresses. Examiner contends on page 3 of the Final Office Action that Kai discloses a plurality of bank pointers for each bank, namely, [(SPM+SPL),$\overline{\text{SPL}}$]. However, as clearly disclosed in Col. 5, line 11-62, for example, SPM merely denotes an address and SPL denotes a least significant bit. Thus, although SPM and SPL may be used to generate addresses that "point" to memory locations in the stack, it is abundantly clear that they do not comprise *stack pointers that store bank address data for respective stack banks*, as essentially claimed in claims 1, 2 and 6. Furthermore, although Koppala may disclose multiple pointers (e.g., Col. 8, lines 33-64), there is nothing in Koppala that discloses or suggests that these pointers comprise bank pointers that point to different banks of the stack memory (store address data for the banks) as claimed.

Therefore, even with Kai and Koppala combined, there is no teaching or suggestion of *a bank pointer associated with each bank of a stack, wherein each bank pointer stores address data for its respective bank*, as essentially claimed in claims 1, 2 and 6. Accordingly, the combination of Kai and Koppala fails to disclose or suggest all the elements of claims 1, 2 and 6.

Moreover, claims 3-5 depend from claim 2 and claims 7-8 depend from claim 6. As such, these claims are believed to be patentable and non-obvious over the combination of Kai and Koppala for at least the reasons given above for claims 2 and 6.

Accordingly, the withdrawal of the rejection under 35 U.S.C. §103(a) is respectfully requested.

Respectfully submitted,



Frank DeRosa
Reg. No. 43,584
Attorney for Applicant(s)

F. Chau & Associates, LLP
1900 Hempstead Tnpk.
East Meadow, NY 11553
TEL.: (516) 357-0091
FAX: (516) 357-0092